实验题目：

+ 对称加密体系具有较为广泛的应用。任写一个文本文件，将其内容用DES、AES方式加密然后解密。

// DES实现加密和解密

import javax.crypto.Cipher; import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException; import javax.crypto.SecretKey;

import java.security.NoSuchAlgorithmException; import java.security.Security;

public class P12\_01

{

//KeyGenerator提供对称密钥生成器的功能，支持各种算法 private KeyGenerator keygen; //SecretKey负责保存对称密钥 private SecretKey deskey; //Cipher负责完成加密或解密工作 private Cipher c; //该字节数组负责保存加密的结果 private byte[] cipherByte;

public P12\_01()

{

Security.addProvider(new com.sun.crypto.provider.SunJCE()); try

{

//实例化支持DES算法的密钥生成器(算法名称命名需按规定，否则抛出异常)

keygen = KeyGenerator.getInstance("DES");

//生成密钥

deskey = keygen.generateKey(); //生成Cipher对象，指定其支持DES算法 c = Cipher.getInstance("DES");

}

catch(NoSuchAlgorithmException ex)

{

ex.printStackTrace();

}

catch(NoSuchPaddingException ex)

{

ex.printStackTrace();

}

}

/\*对字符串str加密\*/ public byte[] createEncryptor(String str)

{

try

{

//根据密钥，对Cipher对象进行初始化,ENCRYPT\_MODE表示加密模式 c.init(Cipher.ENCRYPT\_MODE, deskey);

byte[] src = str.getBytes(); //加密，结果保存进cipherByte cipherByte = c.doFinal(src);

}

catch(java.security.InvalidKeyException ex)

{

ex.printStackTrace();

}

catch(javax.crypto.BadPaddingException ex)

{

ex.printStackTrace();

}

catch(javax.crypto.IllegalBlockSizeException ex)

{

ex.printStackTrace();

}

return cipherByte;

}

/\*对字节数组buff解密\*/

public byte[] createDecryptor(byte[] buff)

{

try

{

//根据密钥，对Cipher对象进行初始化,ENCRYPT\_MODE表示解密模式 c.init(Cipher.DECRYPT\_MODE, deskey); //得到明文，存入cipherByte字符数组 cipherByte = c.doFinal(buff);

}

catch(java.security.InvalidKeyException ex)

{

ex.printStackTrace();

}

catch(javax.crypto.BadPaddingException ex)

{

ex.printStackTrace();

}

catch(javax.crypto.IllegalBlockSizeException ex)

{

ex.printStackTrace();

}

return cipherByte;

}

public static void main(String[] args) throws Exception

{

P12\_01 p12\_01 = new P12\_01(); String msg = "郭克华\_安全编程技术"; System.out.println("明文是：" + msg);

byte[] enc = p12\_01.createEncryptor(msg); System.out.println("密文是：" + new String(enc)); byte[] dec = p12\_01.createDecryptor(enc);

System.out.println("解密后的结果是：" + new String(dec));

}

}

// AES实现加密和解密

import java.security.Security;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

public class P12\_03

{

public static void main(String[] args) throws Exception

{

//KeyGenerator提供对称密钥生成器的功能，支持各种算法

KeyGenerator keygen; //SecretKey负责保存对称密钥 SecretKey deskey; //Cipher负责完成加密或解密工作

Cipher c;

Security.addProvider(new com.sun.crypto.provider.SunJCE()); //实例化支持AES算法的密钥生成器，算法名称用AES keygen = KeyGenerator.getInstance("AES"); //生成密钥 deskey = keygen.generateKey();

//生成Cipher对象，指定其支持AES算法 c = Cipher.getInstance("AES");

String msg = "郭克华\_安全编程技术";

System.out.println("明文是：" + msg);

//根据密钥，对Cipher对象进行初始化,ENCRYPT\_MODE表示加密模式 c.init(Cipher.ENCRYPT\_MODE, deskey);

byte[] src = msg.getBytes(); //加密，结果保存进enc byte[] enc = c.doFinal(src);

System.out.println("密文是：" + new String(enc));

//根据密钥，对Cipher对象进行初始化,ENCRYPT\_MODE表示加密模式 c.init(Cipher.DECRYPT\_MODE, deskey); //解密，结果保存进dec byte[] dec = c.doFinal(enc);

System.out.println("解密后的结果是：" + new String(dec));

}

}

+ 单向加密算法不需要密钥，在数据认证方面具有较为广泛的应用。任写一个文本文件，将其内容用MD5算法进行加密，然后修改这个文本文件，再加密，比较两次的密文。

// MD5实现加密

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class P12\_05

{

// MD5加密

public byte[] encrypt(String msg)

{

try

{

//根据MD5算法生成MessageDigest对象 MessageDigest md5 = MessageDigest.getInstance("MD5"); byte[] srcBytes = msg.getBytes(); //使用srcBytes更新摘要 md5.update(srcBytes); //完成哈希计算,得到result byte[] resultBytes= md5.digest();

return resultBytes;

}

catch(NoSuchAlgorithmException e)

{

e.printStackTrace();

}

return null;

}

public static void main(String[] args)

{

String msg = "郭克华\_安全编程技术"; System.out.println("明文是：" + msg);

P12\_05 p12\_05 = new P12\_05();

byte[] resultBytes = p12\_05.encrypt(msg); String result = new String(resultBytes); System.out.println("密文是：" + result);

}

}

+ 查阅相关文献，寻找一种简单的数字水印生成方法，并实现。

**package** security;

**import** java.awt.\*;

**import** java.awt.image.BufferedImage;

**import** java.io.\*;

**import** javax.imageio.ImageIO;

**public** **class** Watermark{

**public** Watermark(){**super**();};

/\*\*

\* 在源图片上设置水印文字

\*/

**public** **void** WordsToImage(String srcImagePath,**float** alpha,

String font,**int** fontStyle,**int** fontSize,Color color,

String inputWords,**int** x,**int** y,String imageFormat,String toPath) **throws** IOException{

FileOutputStream fos=**null**;

**try** {

//读取图片

BufferedImage image = ImageIO.*read*(**new** File(srcImagePath));

//创建java2D对象

Graphics2D g2d=image.createGraphics();

//用源图像填充背景

g2d.drawImage(image, 0, 0, image.getWidth(), image.getHeight(), **null**, **null**);

//!!!!

AlphaComposite ac = AlphaComposite.*getInstance*(AlphaComposite.***SRC\_OVER***, alpha);

//为 Graphics2D 上下文设置 Composite。 Composite 用于所有绘制方法中，如 drawImage、

//drawString、draw 和 fill。 它指定新的像素如何在呈现过程中与图形设备上的现有像素组合。

g2d.setComposite(ac);

//设置文字字体名称、样式、大小

g2d.setFont(**new** Font(font, fontStyle, fontSize));

g2d.setColor(color);//设置字体颜色

g2d.drawString(inputWords, x, y); //输入水印文字及其起始x、y坐标

g2d.dispose();

//将水印后的图片写入toPath路径中

fos=**new** FileOutputStream(toPath);

ImageIO.*write*(image, imageFormat, fos);

}

//文件操作错误抛出

**catch** (Exception e) {

e.printStackTrace();

}**finally**{

**if**(fos!=**null**){

fos.close();

}

}

}

/\*\*

\* 在源图像上设置图片水印

\*/

**public** **void** ImageToImage(String srcImagePath,String appendImagePath,

**float** alpha,**int** x,**int** y,**int** width,**int** height,

String imageFormat,String toPath) **throws** IOException{

FileOutputStream fos = **null**;

**try** {

//读图

BufferedImage image = ImageIO.*read*(**new** File(srcImagePath));

//创建java2D对象

Graphics2D g2d=image.createGraphics();

//用源图像填充背景

g2d.drawImage(image, 0, 0, image.getWidth(), image.getHeight(), **null**, **null**);

//关键地方

AlphaComposite ac = AlphaComposite.*getInstance*(AlphaComposite.***SRC\_OVER***, alpha);

g2d.setComposite(ac);

BufferedImage appendImage = ImageIO.*read*(**new** File(appendImagePath));

g2d.drawImage(appendImage, x, y, width, height, **null**, **null**);

g2d.dispose();

fos=**new** FileOutputStream(toPath);

ImageIO.*write*(image, imageFormat, fos);

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

**if**(fos!=**null**){

fos.close();

}

}

}

**public** **static** **void** main(String[] args) **throws** Exception{

Watermark imageObj = **new** Watermark();

//源图片路径

String srcImagePath = "F:/1.jpg";

//水印图片路径

String appendImagePath = "F:/logo.jpg";

// ---- 宋体 普通字体 77号字 红色 透明度0.4"

**float** alpha = 0.4F;

String font = "宋体";

**int** fontStyle = Font.***PLAIN***;

**int** fontSize = 40;

Color color = Color.***red***;

String inputWords = "水印啊";

**int** x = 150;

**int** y = 250;

String imageFormat = "jpg";

//水印文字后的存储路径

String wToPath = "F:/2.jpg";

//水印图片后的存储路径

String IToPath = "F:/2s.jpg" ;

imageObj.WordsToImage(srcImagePath, alpha, font, fontStyle,

fontSize, color, inputWords, x, y, imageFormat, wToPath);

imageObj.ImageToImage(srcImagePath, appendImagePath, alpha,

x, y, 150, 250, imageFormat, IToPath);

}

}

+ 编写一个实际的例子，实现SQL注入。

+ 编写一段代码，防范SQL注入。

public static void main(String[] args) throws Exception {

Scanner sc = new Scanner(System.in);

System.out.println("账号：");

String uid = sc.nextLine();

System.out.println("密码：");

String pwd = sc.nextLine();

Class.forName("com.mysql.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb?characterEncoding=GBK","root","");

Statement state = conn.createStatement();

String sql = "select \* from users where user ='"+uid+"' and password ='"+pwd+"' ";

ResultSet rs = state.executeQuery(sql);

boolean ok = rs.next();

if(ok){

System.out.println("欢迎"+rs.getString(3)+"回来");

}

else{

System.out.println("您输入的账号密码错误");

}

conn.close();

}

我们正常输入账号密码是运行正确的，但是当我们账号输入：kjaskj' or 1=1 #  密码输入：klkjl;  就会出现以下的结果：

欢迎张三回来

这里，关键在  账号里面的那个单引号 “ ‘ ’”和后面 or 1=1以及#号（我们这里用的是mysql，oracle后面用 --）。这样查询语句就变成了：

select \* from users where user ='kjaskj'or1=1#' and password ='"+pwd+"'

该双划符号#告诉SQL解析器，右边的东西全部是注释，所以不必理会。这样，查询字符串相当于：

select \* from users where user =''OR1=1. 这样输出的就是ture。 就能不用账号密码直接进入。

public static void main(String[] args) throws Exception {

Scanner sc = new Scanner(System.in);

System.out.println("账号：");

String uid = sc.nextLine();

System.out.println("密码：");

String pwd = sc.nextLine();

Class.forName("com.mysql.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb?characterEncoding=GBK","root","");

Statement state = conn.createStatement();

uid=uid.replace("\'", "\"");//替换单双引号

String sql = "select \* from users where user ='"+uid+"' and password ='"+pwd+"' ";

ResultSet rs = state.executeQuery(sql);

boolean ok = rs.next();

if(ok){

System.out.println("欢迎"+rs.getString(3)+"回来");

}

else{

System.out.println("您输入的账号密码错误");

}

conn.close();

}

+ 编写一个程序，客户输入一个数字，打印其平方。但是如果出错，程序不断提示客户重新输入，直到他输入正确为止。